Mais 202 - Deliverable 2

Alexnder Cui

February 11, 2020

1 Problem statement

In the next couple weeks, I will create a project that aims to translate alphanumeric characters of the american sign language into text.

2 Data pre-processing

I will continue to be working with the same two datasets for my final project taken from Kaggle. For this deliverable however, I restricted myself to using only the dataset presented here: https://www.kaggle.com/grassknoted/asl-alphabet. This is a dataset with approximately 69 000 images and 29 classes. I have chosen this dataset as I believe it is large enough to create a working model. Due to time constraints, I was not able to perform and pre-processing on my data. It would be interesting however, to see if pre-processing images as black and white would yield any better results. Finally, only 10% of my dataset was used. This was done to speed up the training process (although resulting in results that may be less accurate)

3 Machine learning model

For this deliverable, I have maintained my initial proposal and implemented convolutional neural networks. To do this, I used the pre-trained model "Resnet" taken from the PyTorch library. In my final project, I will attempt to train my own model. However, I will need to conduct further research on this topic before I can propose a suitable architecture.

4 Preliminary results

Preliminary results (Figure 1) show that the current model works well at classifying some signs (1:"B", 2:"C",etc..) and not so much for others (4:"D",5:"E",etc...). Despite difficulties classifying certain signs, these result suggests that it would be feasible to translate ASL to text. In the current deliverable, only 10% of the available data was used. If the full dataset was used, I would expect better results. The question of feasibility when incorporating webcam features still remains to be answered.

	precision	recall	f1-score	support
0.0	0.7826	0.7500	0.7660	48
1.0	0.9706	0.6875	0.8049	48
2.0	0.9524	0.8333	0.8889	48
3.0	0.9048	0.7917	0.8444	48
4.0	0.5000	0.8125	0.6190	48
5.0	0.7273	0.8333	0.7767	48
6.0	0.5574	0.7083	0.6239	48
7.0	0.5974	0.9583	0.7360	48
8.0	0.8519	0.4792	0.6133	48
9.0	0.4578	0.7917	0.5802	48
10.0	0.5714	0.9167	0.7040	48
11.0	0.6800	0.7083	0.6939	48
12.0	0.7742	0.5000	0.6076	48
13.0	0.6981	0.7708	0.7327	48
14.0	0.8043	0.7708	0.7872	48
15.0	1.0000	0.6667	0.8000	48
16.0	0.8868	0.9792	0.9307	48
17.0	0.4526	0.8958	0.6014	48
18.0	0.7353	0.5208	0.6098	48
19.0	0.7500	0.5625	0.6429	48
20.0	0.5116	0.4583	0.4835	48
21.0	0.7222	0.2708	0.3939	48
22.0	0.8537	0.7292	0.7865	48
23.0	0.6667	0.3750	0.4800	48
24.0	0.8049	0.6875	0.7416	48
25.0	0.5882	0.4167	0.4878	48
26.0	0.8600	0.8958	0.8776	48
27.0	0.9583	0.9583	0.9583	48
28.0	0.9737	0.7708	0.8605	48
accuracy			0.7069	1392
macro avg	0.7446	0.7069	0.7046	1392
weighted avg	0.7446	0.7069	0.7046	1392

Figure 1: Precision, recall, F1 score

5 Next steps

The next step in this project would be to create and train my own CNN, allowing for greater flexibility and hopefully better accuracy.